Estimating Sequence Similarity from Read Sets for Clustering Sequencing Data

Petr Ryšavý(✉) and Filip Železný

Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic
{rysavpe1,zelezny}@fel.cvut.cz

Abstract. Clustering biological sequences is a central task in bioinformatics. The typical result of new-generation sequencers is a set of short substrings ("reads") of a target sequence, rather than the sequence itself. To cluster sequences given only their read-set representations, one may try to reconstruct each one from the corresponding read set, and then employ conventional (dis)similarity measures such as the edit distance on the assembled sequences. This approach is however problematic and we propose instead to estimate the similarities directly from the read sets. Our approach is based on an adaptation of the Monge-Elkan similarity known from the field of databases. It avoids the NP-hard problem of sequence assembly and in empirical experiments it results in a better approximation of the true sequence similarities and consequently in better clustering, in comparison to the first-assemble-then-cluster approach.

Keywords: Read sets · Similarity · Hierarchical clustering

1 Introduction

Sequencing means reading the sequence of elements that constitute a polymer, such as the DNA. The *human genome project* [5] completed in 2003 was a prime example of sequencing, resulting in the identification of almost the entire genomic sequence (over 3 billion symbols) of a single human. Sequencing becomes technologically difficult as the length of the read sequence grows. The common principle of *new-generation sequencing* (NGS) is that only very short substrings (10's to 100's of symbols) are read at random positions of the sequence of interest. It is usually required that the number of such read substrings, called *reads*, is such that with high probability each position in the sequence is contained in multiple reads; the number of such reads is termed *coverage*. The complete sequence is determined by combinatorial assembly of the substrings guided by their suffix-prefix overlaps. For example, one possible assembly of reads {AGGC, TGGA, GCT} is AGGCTGGA. Short reads imply low cost of wet-lab sequencing traded off with high computational cost of assembly. Indeed the assembly task can be posed as searching the Hamiltonian path in a graph of mutual overlaps.

One of the central tasks in computational biology is to infer phylogenetic trees, which typically amounts to hierarchical clustering of genomes. When they

H. Boström et al. (Eds.): IDA 2016, LNCS 9897, pp. 204–214, 2016.
DOI: 10.1007/978-3-319-46349-0_18

are represented only through sequences read-sets, the bioinformatician is forced to reconstruct the sequences from the read-sets prior to clustering. This of course entails the solution of the NP-hard assembly problem for each data instance with little guarantees regarding the quality of the resulting putative sequence. This motivates the question whether the assembly step could be entirely avoided. We address this question here by proposing a similarity function computable directly on the read sets, that should approximate the true similarities on the original sequences.

Related work includes studies on clustering NGS data (e.g. [1,7]). They however deal with clustering *reads* and we are not aware of a previous attempt to cluster *read-sets*. The paper [16] proposes a similarity measure for NGS data, but again it operates on the level of reads. The previous work [4,13] also aims at avoiding the assembly step in learning from NGS data but these studies concern supervised classification learning and they do not elaborate on read-set similarity.

In the following section, we design the similarity (or, reversely) distance function. Then we provide a brief theoretical analysis of it. In Sect. 4 we compare it to the conventional approach on genomic data and then we conclude the paper.

2 Distance Function Design

The functor $|.|$ will denote the absolute value, cardinality and length (respectively) for a number, set, and string argument. Let $\text{dist}(A, B)$ denote the Levenshtein distance [6] between strings A and B. The function measures the minimum number of edits (insertions, deletions, and substitutions) needed to make the strings identical, and is a typical example of a sequence dissimilarity measure used in bioinformatics. It is a property of the distance that

$$\text{dist}(A, B) \leq \max\{|A|, |B|\}. \tag{1}$$

We will work with constants $l \in \mathbb{N}, \alpha \in \mathbb{R}$ called the *read length* and *coverage*, respectively, which are specific to a particular sequencing experiment. A *read-set* R_A of string A such that

$$|A| \gg l \tag{2}$$

is a multiset of[1]

$$|R_A| = \frac{\alpha}{l}|A| \tag{3}$$

substrings sampled i.i.d. with replacement from the uniform distribution on all the $|A| - l + 1$ substrings of length l of A. Informally, the coverage α indicates the average number of reads covering a given place in A.

Our goal is to propose a distance function $\text{Dist}(R_A, R_B)$ that approximates $\text{dist}(A, B)$ for read-sets R_A and R_B of arbitrary strings A and B. We also want $\text{Dist}(R_A, R_B)$ to be more accurate and less complex to calculate than a natural estimate $\text{dist}(\hat{A}, \hat{B})$ in which the arguments represent putative sequences reconstructed from R_A and R_B using *assembly algorithms* such as [3,11,15].

[1] Should the right hand side be non-integer, we neglect its fractional part.

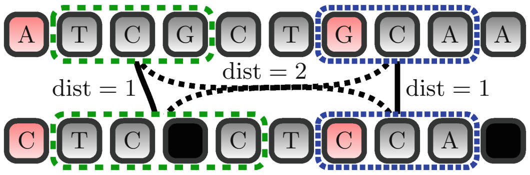

Fig. 1. We calculate read-read distances in order to find matching pairs of reads. For each read from the first sequence we find the least distant read in the second sequence. We see optimal alignment of ATCGCTGCAA and CTCCTCCA. Read TCG is paired with TCC.

2.1 Base Case: Which Reads Belong Together

A natural approach to instantiate $\text{Dist}(R_A, R_B)$ is to exploit the $|R_A||R_B|$ pairwise Levenshtein distances between the reads in R_A and R_B. Most of those values are useless because they match reads from completely different parts of sequences A and B. Therefore we want to account only for those pairs which likely belong together.

If we seek a read from R_B that matches a read $a_i \in R_A$, we make the assumption that the most similar read $b_j \in R_B$ is the one that we look for (see Fig. 1), i.e.,

$$b_j = \underset{b_k \in R_B}{\arg\min}\, \text{dist}(a_i, b_k).$$

To calculate the distance from R_A to R_B, we average over all reads from R_A:

$$\underset{\mathsf{ME}}{\text{Dist}}(R_A, R_B) = \frac{1}{|R_A|} \sum_{a_i \in R_A} \min_{b_j \in R_B} \text{dist}(a_i, b_j). \tag{4}$$

This idea was presented in [8] for searching duplicates in database systems. The method is known as the *Monge-Elkan similarity*[2] (hence the ME label) and entails a simple but effective approximation algorithm.

$\text{Dist}_{\mathsf{ME}}(R_A, R_B)$ is non-symmetric in general, which is undesirable given that the approximated distance $\text{dist}(A, B)$ is known to be symmetric. Therefore we define a symmetric version by averaging both directions

$$\underset{\mathsf{MES}}{\text{Dist}}(R_A, R_B) = \frac{1}{2}\left(\underset{\mathsf{ME}}{\text{Dist}}(R_A, R_B) + \underset{\mathsf{ME}}{\text{Dist}}(R_B, R_A)\right). \tag{5}$$

2.2 Distance Scale

Consider duplicating a non-empty string A into AA and assume $R_{AA} = R_A \cup R_A$. Typically for a B similar to A we expect that $\text{dist}(AA, B) > \text{dist}(A, B)$ but the

[2] Here we alter the Monge-Elkan similarity into a distance measure. The standard way of using Monge-Elkan is as a similarity measure with min replaced by max and distance calculation by similarity calculation.

(symmetric) Monge-Elkan distance will not change, i.e. $\mathrm{Dist}_{\mathsf{MES}}(R_{AA}, R_B) = \mathrm{Dist}_{\mathsf{MES}}(R_A, R_B)$, indicating a discrepancy that should be rectified.

In fact, $\mathrm{Dist}_{\mathsf{MES}}$ has the constant upper bound l, which is because it is the average (c.f. (4) and (5)) of numbers no greater than l (see (1)). On the other hand, $\mathrm{dist}(A, B)$ has a non-constant upper bound $\max\{|A|, |B|\}$ as by (1).

To bring $\mathrm{Dist}_{\mathsf{MES}}(A, B)$ on the same scale as $\mathrm{dist}(A, B)$, we should therefore multiply it by the factor $\max\{|A|, |B|\}/l = \max\{|A|/l, |B|/l\}$. By (3) we have $|A| = \frac{l}{\alpha}|R_A|$, yielding the factor $\max\{|R_A|/\alpha, |R_B|/\alpha\}$, in which α is a constant divisor which can be neglected in a distance function. Therefore, we modify the read distance into

$$\underset{\mathsf{MESS}}{\mathrm{Dist}} = \max\{|R_A|, |R_B|\} \underset{\mathsf{MES}}{\mathrm{Dist}}\,. \tag{6}$$

2.3 Margin Gaps

Consider the situation in Fig. 2 showing two identical sequences each with one shown read. The Levenshtein distance between the two reads is non-zero due to the one-symbol trailing (leading, respectively) gap of the top (bottom) read caused only by the different random positions of the reads rather than due to a mismatch between the sequences. Thus there is an intuitive reason to pardon margin gaps up to certain size t

$$t < \frac{l}{2} \tag{7}$$

when matching reads. Here, t should not be too large as otherwise the distance could be nullified for pairs of long reads with small prefix-suffix overlaps, which would not make sense.

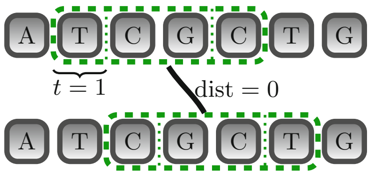

Fig. 2. Because reads locations in sequences are random, we do not want to penalize small leading or trailing gaps.

To estimate a good value for t, consider sequence A and its sampled read-set R_A. We now sample an additional read a of length l from A. Ideally, there should be a zero-penalty match for a in R_A as a was sampled from the same sequence as R_A was. This happens iff there is a read in R_A sampled from the same position in A as a, or from a position shifted by up to t symbols to the left or right as then the induced gaps are penalty-free. Since R_A is an uniform-probability i.i.d. sample from A, the probability that a given read from R_A starts at one of these $1+2t$ positions is[3] $\frac{1+2t}{|A|}$. We want to put an upper bound $\varepsilon > 0$ on the probability that this happens for none of the $|R_A| = \frac{\alpha}{l}|A|$ reads in R_A:

$$p = \left(1 - \frac{1+2t}{|A|}\right)^{\frac{|A|\cdot\alpha}{l}} \leq \varepsilon \ .$$

Consider the first-order Taylor approximation $(1+x)^n = 1 + nx + \varepsilon'$ where the difference term $\varepsilon' > 0$ decreases with decreasing $|x|$. Due to Ineq. (7) and (2), $\frac{1+2t}{|A|}$ is small and we can apply the approximation on the above formula for p, yielding

$$p = 1 - \frac{2t+1}{|A|}\frac{|A|\cdot\alpha}{l} + \varepsilon' = 1 - (2t+1)\frac{\alpha}{l} + \varepsilon' \leq \varepsilon \ .$$

For simplicity, we choose $\varepsilon = \varepsilon'$. The smallest gap size t for which the inequality is satisfied is obtained by solving $1 - (2t+1)\frac{\alpha}{l} = 0$, yielding

$$t = \frac{1}{2}\left(\frac{l}{\alpha} - 1\right) .$$

This choice of t matches intuition in that with larger read-length l we can allow a larger grace gap t but with larger coverage α, t needs not be so large as there is a higher chance of having a suitably positioned read in the read-set. Another way to look at it is to realize that reads in a read-set are approximately $\frac{l}{\alpha}$ positions from each other. Consider matching read a to reads from R_A. If there is a read $a_1 \in R_A$ requiring gap larger than $\frac{l}{2\alpha}$ to match a, then there will typically be another read $a_2 \in R_A$ requiring gap at most $\frac{l}{2\alpha}$ (see Fig. 3).

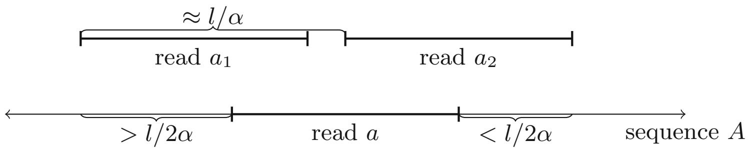

Fig. 3. Illustration to reasoning in Sect. 2.3

We implemented the grace margin gaps into a further version Dist$_{\mathsf{MESSG}}$ of the constructed distance function, which required only a small change to the standard Wagner-Fischer algorithm [14].[4] When the algorithm is filling the first

[3] Strictly speaking, this reasoning is incorrect if read a is drawn from a place close to A's margins, more precisely, if it starts in fewer than t ($t+l$, respectively) symbols from A's left (right) margin, as then not all of the $2t$ shifts are possible. This is however negligible due to Ineq. (2).

[4] The dynamic programming algorithm for calculating the Levenshtein distance [6] is commonly called Wagner-Fischer algorithm [14]. When we refer to sequence alignment problem in bioinformatics, this algorithm is often called Needleman-Wunsch algorithm [9].

or the last row and column of the table, margin gaps up to t symbols are not penalized. Larger margin gaps are penalized in a way that satisfies the constraint that the distance between a word a and an empty word is $|a|$. In particular, the standard linear gap penalty is replaced with a piecewise linear function that gives cost of margin gap at x-th position

$$g(x) = \begin{cases} 0, & \text{if } 0 \leq x \leq t-1, \\ 2\frac{x-t+1}{l+1-2t}, & \text{if } t-1 < x \leq l-t, \\ 2, & \text{if } l-t < x < l. \end{cases} \tag{8}$$

2.4 Missing Read

Sometimes there is no good match for read a_i in R_B. During evolution the substring that contained a_i may have been inserted into A or may have vanished from B. Therefore if

$$\text{dist}(a_i, b_j) \geq \theta$$

for some reads a_i and b_j and threshold θ, we consider a_i and b_j to be dissimilar and we force their distance to be l (See Fig. 4).

Threshold θ should be a linear factor of the maximal distance between two sequences of length l, i.e. $\theta = \theta' \cdot l$. Value of θ' should reflect the probability that the read is in one sequence and not in the other. Because the true probability is hidden, it needs to be determined empirically.

The distance function equipped with the missing read detection as described gives rise to the last version denoted as $\text{Dist}_{\mathsf{MESSGM}}$.

3 Theoretical Analysis

3.1 Asymptotic Complexity

Calculating $\text{dist}(A, B)$ for sequences A and B requires $\Theta(|A||B|)$ operations if we use the standard Wagner-Fischer dynamic programming algorithm [14]. This algorithm also requires $\Theta(\min(|A|, |B|))$ memory as we are interested only in distance and not in the alignment. To calculate $\text{Dist}_{\mathsf{ME}}$ we need to know the distances between all pairs of reads, so we have to evaluate (see (3)) $\frac{\alpha}{l}|A|\frac{\alpha}{l}|B|$ distances where each one requires l^2 operations. Therefore $\alpha^2|A||B|$ operations are required. For the symmetric version $\text{Dist}_{\mathsf{MES}}$ we make $2\alpha^2|A||B|$ operations, which can be reduced to $\alpha^2|A||B|$ operations and $\Theta(l + \frac{\alpha}{l}(|A| + |B|))$ memory. Further modifications (MESS, MESSG, MESSGM) do not change the asymptotic complexity.

The constants α and l are determined by the sequencing technology and the independent complexity factors are $|A|$ and $|B|$. To calculate the distance in the conventional way as $\text{dist}(\hat{A}, \hat{B})$ requires to reconstruct $\hat{A}$ and $\hat{B}$ from the respective read-sets through an assembly algorithm. This is an NP-hard problem which becomes non-tractable for large $|A|$ and $|B|$, and which is avoided by our approach.

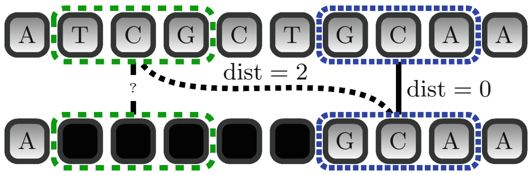

Fig. 4. If the distance between a read and its closest counterpart is greater than threshold θ, we assume that the read matches to a gap in the sequence alignment.

3.2 Metric Properties

$\text{Dist}_{\mathsf{MES}}$ as well as the later versions are all symmetric and non-negative but none of the proposed versions satisfies the identity condition ($\text{dist}(a, b) = 0$ iff $a = b$) or the triangle inequality, despite being based on the Levenshtein distance dist, which is a metric. For example, let $R_A = \{\mathsf{ATC}, \mathsf{ATC}, \mathsf{GGG}\}$, let $R_B = \{\mathsf{ATA}, \mathsf{GGG}\}$, and let $R_C = \{\mathsf{CTA}, \mathsf{GGG}\}$. Then $\text{Dist}_{\mathsf{MES}}(R_A, R_B) = \frac{7}{12}$, and $\text{Dist}_{\mathsf{MES}}(R_B, R_C) = \frac{1}{2}$ but $\text{Dist}_{\mathsf{MES}}(R_A, R_C) = \frac{14}{12} > \frac{7}{12} + \frac{1}{2}$. While this might lead to counter-intuitive behavior of the proposed distances in certain applications, the violated conditions are not requirements assumed by clustering algorithms.

4 Experimental Evaluation

The **purpose** of the experiments is to compare different methods for estimating the Levenhstein distance $\text{dist}(A, B)$ for various real DNA sequences A, B from their read sets R_A, R_B. The methods include (i) our newly proposed distances (MES, MESS, MESSG, MESSGM) applicable directly on R_A, R_B and implemented in Java with maximum of shared code,[5] (ii) the conventional method based on assembling estimates $\hat{A}, \hat{B}$ of the original sequences A, B using 3 common de-novo gene assemblers (ABySS [11], edena [3] and SSAKE [15]) and then estimating $\text{dist}(A, B)$ as $\text{dist}(\hat{A}, \hat{B})$, (iii) a trivial baseline method estimating $\text{dist}(A, B)$ as $\max\{|R_A|, |R_B|\}$. All the 3 assembly algorithms were configured with the default parameters and the current official C++ version was used. When a result of an assembly procedure consisted of multiple contigs, we selected the longest one.

The evaluation **criteria** consist of (i) the Pearson's correlation coefficient measuring the similarity of the distance matrix produced by the respective methods to the true distance matrix, (ii) The Fowlkes-Mallows index [2] measuring the similarity between the tree produced by a hierarchical clustering algorithm using the true distance matrix, and the tree induced from a distance matrix estimated by the respective method, (iii) runtime needed for assembly (if applicable), distance matrix calculation, and clustering time. For hierarchical clustering, we used the UPGMA algorithm [12] and the neighbor-joining algorithm [10].

[5] Implementation is available on https://github.com/petrrysavy/readsIDA2016.

The Fowlkes-Mallows index shows how much the resulting trees differ in structure. Both trees are first cut into k clusters for $k = 2, 3, \ldots, n-1$. Then the clusterings are compared based on the number of common objects among each pair of clusters. By this procedure we obtain a set of values B_k that show how much the trees differ at various levels.

The testing **data** contain two datasets. The first dataset[6] contains 12 influenza virus genome sequences plus an outgroup sequence. The second dataset[7] contains 17 genomes of different viruses. Furthermore, we used an independent third *training* dataset[8] to tune the value of θ' (see Sect. 2.4). All the sequences were downloaded from the ENA repository http://www.ebi.ac.uk/ena.

In the preliminary **tuning** experiment, the value $\theta' = 0.35$ achieved the best Pearson's correlation coefficient on the training dataset and we carried this value over to the testing experiments. Out of curiosity, we also tried to optimize θ' on the testing datasets obtaining similar values (around 0.3), indicating relative stability of this parameter.

The main experimental **results** are shown in Table 1. The four partitions of the table correspond to the two datasets ($N = 13$, $N = 17$), each used twice with different settings of sequencing coverage α and read length l. The Pearson's correlation coefficient (column 'corr.') demonstrates clear dominance of the MESSG and MESSGM methods (Sects. 2.3 and 2.4), which are the most developed versions of our approach. The MESSG differs from MESSGM only by not discarding poorly matching reads. This finding is generally supported also by the Fowlkes-Mallows index (last four columns) shown for two levels of trees learned by two methods. Figure 5 provides a more detailed insight into the Fowlkes-Mallows values graphically for all the tree levels. One more (rather surprising) observation is that distance estimates achieved by first assembling sequences from read-sets (last 3 lines in every table partition) are systematically worse than the trivial estimate based just on the read-set sizes (first line in every partition).

Columns 1–5 of Table 1 indicate that all the variants of our approach were systematically slower in terms of absolute runtime than the approaches based on sequence assembly, despite the NP-hard complexity of the latter task. The numbers also show that our asymptotic complexity estimate in Sect. 3.1 is generally correct: the ratio between the time spent on calculating the distances on one hand, and the runtime of the reference method on the other hand, is approximately α^2.

[6] AF389115, AF389119, AY260942, AY260945, AY260949, AY260955, CY011131, CY011135, CY011143, HE584750, J02147, K00423 and outgroup AM050555. The genomes are available at http://www.ebi.ac.uk/ena/data/view/accession.

[7] AB073912, AB236320, AM050555, D13784, EU376394, FJ560719, GU076451, JN680353, JN998607, M14707, U06714, U46935, U66304, U81989, X05817, Y13051 and outgroup AY884005.

[8] CY011119, CY011127, CY011140, FJ966081, AF144300, AF144300, J02057, AJ437618, FR717138, FJ869909, L00163, KJ938716, KP202150, D00664, HM590588, KM874295, $\alpha = 4$, $l = 40$.

Table 1. Runtime, Pearson's correlation coefficient between distance matrices and Fowlkes-Mallows index for $k = 4$ and $k = 8$. The 'reference' method calculates distances from the original sequences.

	Method	assem.	distances	UPGMA	NJ	corr.	UPGMA	UPGMA	NJ	NJ
		ms	ms	ms	ms		B_4	B_8	B_4	B_8
$\alpha = 3,\ l = 30,\ N = 13$	Reference	0	1,587	1	39	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	0	0	1	16	.802	.66	.32	.66	.32
	Dist_{MES}	0	18,192	0	8	.83	.36	.67	.4	1
	$\text{Dist}_{\text{MESS}}$	0	17,132	1	11	.944	1	1	1	1
	$\text{Dist}_{\text{MESSG}}$	0	35,107	0	7	.99	1	1	1	1
	$\text{Dist}_{\text{MESSGM}}$	0	34,911	1	5	.991	1	1	1	1
	ABySS [11]	22,231	7	0	3	.376	.36	.11	.31	.14
	edena [3]	3,501	6	1	7	.404	.36	.12	.31	0
	SSAKE [15]	6,811	1	1	5	.548	.27	.12	.43	0
$\alpha = 3,\ l = 30,\ N = 17$	Reference	0	23,367	1	38	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	0	1	1	17	.902	.67	.66	.85	.86
	Dist_{MES}	0	279,965	0	12	.605	.35	.52	.41	.29
	$\text{Dist}_{\text{MESS}}$	0	279,008	1	16	.935	.67	.92	.85	.94
	$\text{Dist}_{\text{MESSG}}$	0	508,947	1	7	.945	.62	.92	1	.96
	$\text{Dist}_{\text{MESSGM}}$	0	546,985	1	15	.95	.62	.92	1	.96
	ABySS [11]	30,974	16	0	12	.684	.58	.72	.48	.13
	edena [3]	6,287	11	1	91	.666	.58	.63	.46	.13
	SSAKE [15]	12,745	2	23	20	.611	.62	.31	.47	.12
$\alpha = 5,\ l = 100,\ N = 13$	Reference	0	1,653	1	19	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	0	1	0	11	.802	.66	.32	.66	.32
	Dist_{MES}	0	47,238	1	7	.881	.36	.77	.54	1
	$\text{Dist}_{\text{MESS}}$	0	47,703	1	7	.975	1	1	1	1
	$\text{Dist}_{\text{MESSG}}$	0	83,186	0	6	.993	1	1	1	1
	$\text{Dist}_{\text{MESSGM}}$	0	82,973	1	6	.99	1	1	1	.77
	ABySS [11]	29,326	814	0	10	.639	.44	.62	.46	.77
	edena [3]	6,455	128	1	9	.388	.3	.11	.39	.11
	SSAKE [15]	7,258	94	1	7	.706	.54	.21	.72	.11
$\alpha = 5,\ l = 100,\ N = 17$	Reference	0	24,612	1	13	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	0	0	1	16	.903	.67	.66	.85	.86
	Dist_{MES}	0	687,503	1	13	.648	.35	.52	.41	.37
	$\text{Dist}_{\text{MESS}}$	0	680,522	0	19	.935	.67	.86	1	1
	$\text{Dist}_{\text{MESSG}}$	0	1,254,231	4	6	.94	.62	1	1	1
	$\text{Dist}_{\text{MESSGM}}$	0	1,156,072	1	13	.938	.62	1	.69	.94
	ABySS [11]	30,598	5,891	1	12	.576	.51	.66	.47	.15
	edena [3]	9,918	363	2	19	.473	.5	.3	.54	.22
	SSAKE [15]	28,590	374	1	23	.519	.48	.3	.4	.34

5 Conclusions and Future Work

We have proposed and evaluated several variants of a method for estimating edit distances between sequences only from read-sets sampled from them. In experiments, our approach produced better estimates than a conventional approach based on first estimating the sequences themselves by applying assembly algorithms on the read-sets.

A further observation was that the conventional approach was surprisingly fast despite involving the NP-hard assembly problem, and resulted in surprisingly low-quality estimates. A possible explanation for this is that the assemblers

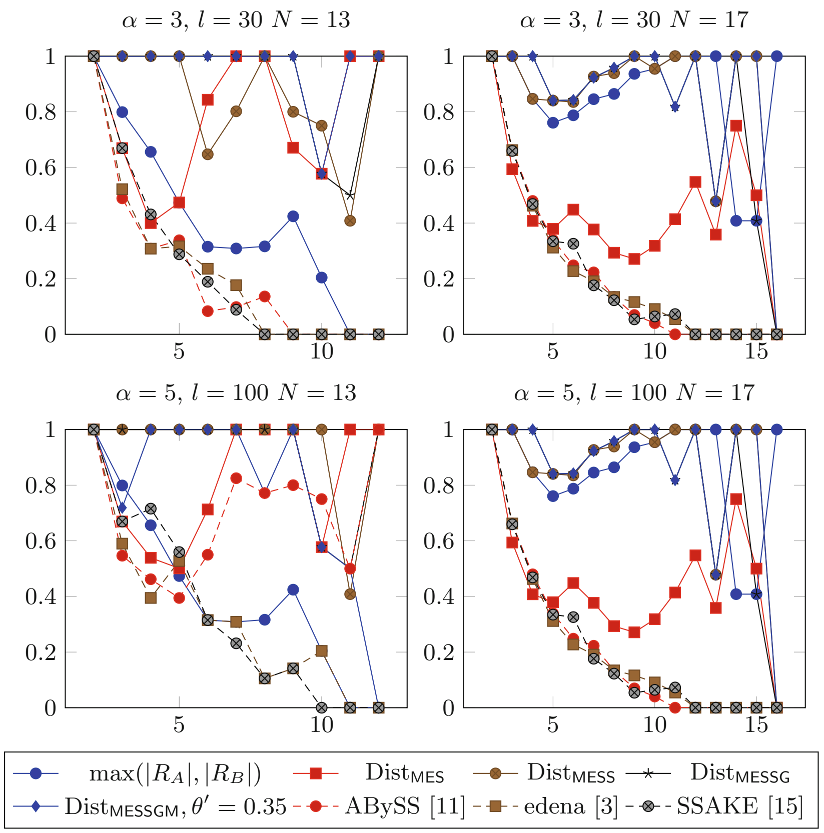

Fig. 5. Plots of Fowlkes-Mallows index B_k versus k. The index compares trees generated by the neighbor-joining algorithm. The tree is compared with the tree generated from the original sequences. If all values are equal to 1, the structures of the trees are the same.

require a higher coverage α to produce good sequences compared to the values we chose ($\alpha = 3, \alpha = 5$).

In our opinion, the most urgent goal for follow-up work is to conduct a more thorough experimental evaluation, and specifically, find the sequences lengths for which the exponential runtime of the assembly-based approach reaches the quadratic runtime of our approach, and then re-evaluate the approximation qualities. Furthermore, our approach offers many directions for technical improvements. For example, one may consider a *partial assembly* approach, in which sets of a few (up to a constant) reads would be pre-assembled and the Monge-Elkan distance would be applied on such partial assemblies. This view would open a 'continuous' spectrum between our approach on one hand, and the conventional assembly-based approach, on which the optimal trade-off could be identified.

Acknowledgment. This work was supported by Czech Science Foundation project 14-21421S.

References

1. Bao, E., Jiang, T., Kaloshian, I., Girke, T.: Seed: efficient clustering of next-generation sequences. Bioinformatics **27**(18), 2502–2509 (2011)
2. Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings. J. Am. Stat. Assoc. **78**(383), 553–569 (1983)
3. Hernandez, D., et al.: De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res. **18**(5), 802–809 (2008)
4. Jalovec, K., Železný, F.: Binary classification of metagenomic samples using discriminative dna superstrings. In: 8th International Workshop on Machine Learning in Systems Biology, MLSB 2014 (2014)
5. Lander, E.: Initial impact of the sequencing of the human genome. Nature **470**(7333), 187–197 (2011)
6. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)
7. Malhotra, R., Elleder, D., Bao, L., Hunter, D.R., Acharya, R., Poss, M.: Clustering pipeline for determining consensus sequences in targeted next-generation sequencing. arXiv (Conrell University Library) arXiv:1410.1608 (2016)
8. Monge, A.E., Elkan, C.P.: The webfind tool for finding scientific papers over the worldwide web. In: Proceedings of the 3rd International Congress on Computer Science Research (1996)
9. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**(3), 443–453 (1970)
10. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**(4), 406–425 (1987)
11. Simpson, J.T., et al.: ABySS: a parallel assembler for short read sequence data. Genome Res. **9**(6), 1117–1123 (2009)
12. Sokal, R.R., Michener, C.D.: A statistical method for evaluating systematic relationships. Univ. Kansas Sci. Bull. **38**, 1409–1438 (1958)
13. Železný, F., Jalovec, K., Tolar, J.: Learning meets sequencing: a generality framework for read-sets. In: 24th International Conference on Inductive Logic Programming, Late-Breaking Papers, ILP 2014 (2014)
14. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. J. ACM **21**(1), 168–173 (1974). http://doi.acm.org/10.1145/321796.321811
15. Warren, R.L., et al.: Assembling millions of short DNA sequences using SSAKE. Bioinformatics **23**(4), 500–501 (2007)
16. Weitschek, E., Santoni, D., Fiscon, G., Cola, M.C.D., Bertolazzi, P., Felici, G.: Next generation sequencing reads comparison with an alignment-free distance. BMC Res. Notes **7**(1), 869 (2014)